API Endpoints

Login and Signup :

Signup: Users will be able to signup for a new account with name, email and password.

endPoint: /userSignup
Type: POST
Success: 200
Failure: 400,409 (User exists)
Input:
{
   data: user_email, name, password
}

Login: Users will be able to login with the email id and password. Prevents login if users email id or password is not registered.

endPoint: /userLogin
Type: GET
Success: 200
Failure: 400,401
Input:
{
  data: user_email,password
 }

My Profile Page

Users should be able to add and update their name, gender, location, description, password, profile picture, a list of topics they are interested in (user should be able to add/edit/delete topic list)

Edit User Details:

endPoint: /editUser/{UserName}
Type: PUT
Success: 200
Failure: 400

Input:
```
{
   Name : “User1”,
 Gender:””,
  Location: “”,
  Description: “”,
  Password:” “,
  Profilepic :Url,
  Topics:[‘Topic1”,”Topic2”,”Topic3”]
}
```

My Communities Page

Create community:

endPoint: /createCommunity
Type: POST
Success: 200
Failure: 400, 409 (community already exists)
Input:
```
{
    communityName: “My community”,
    communityDescription: “This is my community”,
    communityRules: [“rule1”, “rule2”, “rule3”],
    Images: [“image1”, “image2”]
    ….
}
```

Edit community:

endPoint: /editCommunity/{communityName}
Type: PUT
Success: 200
Failure: 400
Input:
```
{
  communityDescription: “This is my community”,
  communityRules: [“rule1”, “rule2”, “rule3”],
   Images: [“image3”, “image4”]
}
```

List Communities:

endPoint: /listCommunities/{communityName}/{pageStart}/{pageEnd}
Type: GET
Success: 200
Failure: 400
Output:

```
[{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
},
{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
},
]
```

Delete Community:

endPoint: /deleteCommunity/{communityName}
Type: DELETE
Success: 200
Failure: 400

MY COMMUNITIES MODERATION PAGE

Functionalities identified:
Show the list of communities created by the user.
API Endpoint : /communitiesCreated/ :user
Type: GET
Success: 200 (OK)

Failure: 400
Response Data:
```
[{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
},
{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
},
]
```

Each community should show the number of users who have requested to join. Clicking on the community should show the list of users with their profile picture name etc in a modal window.
API Endpoint: /getallUserRequest/:community
Type: GET
Success: 200 (OK)
Failure: 400

Response Data:
```
[{
  userName: “Jane Doe”,
  userEmail: “janed@gmail.com”
  gender: ” female”
  profilePhoto: “http:// amazons3.com/…. ”,
  userDescription: “user description 1”,
  listOfTopics: [“topic1”,”topic2”,...],
  location: ” USA ( or we could use latitude/longitude)”
  listOfCommunitiesReq: [ “community1”, “community2”,....],
  listOfCommunitiesAcc: [ “community3”, “community4”,....]



},
{
   userName: “John Doe”,
  userEmail: “johnd@gmail.com”
  gender: ” male”
  profilePhoto: “http:// amazons3.com/…. ”,
  userDescription: “user description 2”,
  listOfTopics: [“topic1”,”topic2”,...],
  location: ” USA ( or we could use latitude/longitude)”,
  listOfCommunitiesReq: [ “community1”, “community2”,....],
  listOfCommunitiesAcc: [ “community3”, “community4”,....],
},
]
```

The community owner should be able to select one or more users (use a checkbox) and approve their request to join the group.
API Endpoint: /acceptRequest (data:{user id, community id/name}) (Move value from one array to another in user document).
Type: POST
Success: 200 (OK)
Failure: 400

Show the list of users who have joined the communities (created by the user) along with their profile picture.
API Endpoint: /getallUser/:community
Type: GET
Success: 200 (OK)
Failure: 400

Response Data: (Query to be written in such a way that out of the entire user data only this is fetched.

```
[{
  userName: “Jane Doe”,
  userEmail: “janed@gmail.com”
   profilePhoto: “http:// amazons3.com/…. ,

},
{
   userName: “John Doe”,
  userEmail: “johnd@gmail.com”
  profilePhoto: “http:// amazons3.com/…. ”,
 },
]
```

The owner should be able to click on a user, display the list of communities they belong to in a modal window and select one or more communities and remove them.

APIs:
Type: GET
Endpoint: /communities/ :user (we can fetch from user document which will have reference to community document)
Success: 200 (OK)
Failure: 400
Response Data:

```
[{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
},
{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
},
]
```

Type: POST/DELETE
Endpoint: /deleteUser (data: user id, community_id)
Success: 200 (OK)
Failure: 400

Community search page:

Communities search pages should show the list of all the communities with most recently created first, with pagination
(Here fetching all the communities from the community table in the database sorting in backend by timestamp (most recent should show up)).

endPoint: /listCommunities//{pageStart}/{pageEnd}
Type: GET
Success: 200
Failure: 400
Output:
```
[{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
}
]
```

Users should be able to upvote or downvote a community
(Community table will have column vote which will get updated here)
(If upvote set column = 1 if downvote set column = 2)
endPoint: /voteCommunities
Type: POST
Success: 200
Failure: 400
Input communityName and email id of user
Action: Query to update the column vote in community table
Response: Column updated

Users should be able to sort communities by created at, most number of users, most

number of posts, most upvoted posts in ascending and descending order.
Sort Communities: We need to sort communities based on the criteria the user has selected. We can send the sorting technique to the backend and display the sorted posts.

endPoint: /sortCommunity
Type: GET
Success: 200
Failure: 400
Input:
```
{
            Sorting_technique: technique_1
            ….
}
```

Users should be able to search for a community by name.

Community Home Page

EndPoint: /getCommunity/{communityName}
Type: GET
Success: 200
Failure: 400
Output:
```
{
  communityName: “community 1”,
  Timestamp: “2021-04-01 5:00 pm”
  communityImages: [“image 1”, “image 2”, “image 3”],
  description: “community description”,
  numberOfPosts: 10,
  numberOfUsers: 100,
},
```
EndPoint: /joinCommunity
Type: POST

Success: 200
Failure: 400
Req Body:
{
name: “community name”
}

EndPoint: /createNewPost
Type: POST
Success: 200
Failure: 400
Req Body:
{
post_type: “text”,
title:”Hello World!!”,
message:”I am alive”
}

EndPoint: /postUpvote
Type: POST
Success: 200
Failure: 400
Req Body:
{
postId:24,
}

EndPoint: /postDownvote
Type: POST
Success: 200
Failure: 400
Req Body:
{
postId:24,
}

EndPoint: /addComment
Type: POST
Success: 200
Failure: 400
Req Body:

```
{
parentId:241
userId: 1234
}
```

Output:

```
{
commentId:241,
parentId:12
}
```

endPoint: /listComments/:parentId
Type: GET
Success: 200
Failure: 400
Output:

```
[
{
commentId:12,
commentData: “Welcome to reddit!”
}
]
```

endPoint: /commentUpvote
Type: POST
Success: 200
Failure: 400
Req Body:

```
{
commentid:24,
}
```

endPoint: /commentDownvote
Type: POST

Success: 200
Failure: 400
Req Body:
```
{
commentid:24,
}
```

My Dashboard:

Show all posts: Get all the communities the user is part of and then posts of those communities.
(We need to store data in such a way that a community collection will have all it's posts and each post will have its own comments stored in the same collection).

endPoint: /getAllPosts
Type: GET
Success: 200
Failure: 401 (Unauthorized)
Input:
```
{
   User: user_id/userEmail
}
```

Sort Posts: We need to sort posts for a user based on the criteria he has selected. We can send the sorting technique to the backend and display the sorted posts.

endPoint: /sortPost
Type: GET
Success: 200
Failure: 401 (Unauthorized)
Input:
```
{
  User: user_id/userEmail,
  Sorting_technique: technique_1
   ....
}
```

My Communities Analytics :

Number of users in a community: Get total users in community by using count in query
Number of posts in a community: Get count of posts of that community
The most upvoted post: Get the post with MAX upvote query
User who created the maximum number of posts: Get the user by counting the user id who created the post
Community with the maximum number of posts: Get this by counting the total posts of communities and getting the MAX of it.

endPoint: /communityAnalytics
Type: GET
Success: 200
Failure: 400
Input:

```
{
  data: community_name/community_id
 }
```

Output:

```
[{
  numberOfUsersInCommunity: “community name”,
  totalPosts: 55,
  mostUpvotedPost: “Post Name”,
  userNameWithMaxPost: “user name”,
  communityWithMaxPosts: “community name”,
}]
```

Invitations to join Communities:

Any user will be able to invite other users to his community only when he is the admin of that community.
So first, we will check all communities and their admins and then only provide him the privilege of inviting other users to the community.
We will require 2 API’s in this case:

1. **One when the community admin invites the other users of reddit**
2. **When the invited user will accept the invite.**

- **The API's can be found below:**

endPoint: /sendInvite
Type: POST
Success: 200
Failure: 400
Input:
{"user@gmail.com"}
Action: Query database to get list of all users on reddit and pass the list to the UI
Output:
{
listOfUsers:
[{"john@gmail.com",
"tim@gmail.com",
"tom@gmail.com"
}]
}
endPoint: /acceptInvite
Type: POST
Success: 200
Failure: 400
Input: {"user@gmail.com"}
Action: Query to get list of all communities where the user has been invited
Response:
[{
"Community1",
"Community2"
}]

My messages:

Endpoint: /searchUsers
Type: GET
Success: 200
Failure: 400
Input:
{
  searchString: The string entered the by the User
 }
Output:
An array of Users like the search string.

Endpoint: /getChat
Type: GET
Success: 200
Failure: 400
Input:
{
   user1: Present Logged in user,
   user2: User the present logged in user wants to chat with
 }
Output:
Whole chat between both the users.

User Profile Page

endPoint: /getUser/{UserName}
Type: GET
Success: 200
Failure: 400
Input: {UserName}
Output:
{
   Name : “User1”,
 Gender:””,
  Location: “”,
  Description: “”,
  Password:” “,
  Profilepic :Url,
  Topics:[‘Topic1”,”Topic2”,”Topic3”]

}